TASK-4

\*Task-4 \*

Perform task-3 with an additional feature to be added that is NAT Gateway to provide the internet access to instances running in the private subnet.

Performing the following steps:

1. Write an Infrastructure as code using terraform, which automatically create a VPC.

2. In that VPC we have to create 2 subnets:

1. public subnet [ Accessible for Public World! ]

2. private subnet [ Restricted for Public World! ]

3. Create a public facing internet gateway for connect our VPC/Network to the internet world and attach this gateway to our VPC.

4. Create a routing table for Internet gateway so that instance can connect to outside world, update and associate it with public subnet.

5. Create a NAT gateway for connect our VPC/Network to the internet world and attach this gateway to our VPC in the public network

6. Update the routing table of the private subnet, so that to access the internet it uses the nat gateway created in the public subnet

7. Launch an ec2 instance which has Wordpress setup already having the security group allowing port 80 sothat our client can connect to our wordpress site. Also attach the key to instance for further login into it.

8. Launch an ec2 instance which has MYSQL setup already with security group allowing port 3306 in private subnet so that our wordpress vm can connect with the same. Also attach the key with the same.

REPO PATH :- <https://github.com/Shailly29/Hybrid-task4.git>

**What is NAT Gateways?**

NAT stands for Network Address Translation.

If you want your EC2 instance in a private subnet can access the internet, this can be achieved only when it can communicate to the internet. However, we do not want to make a subnet public as we want to maintain the degree of control. To overcome the problem, we need to create either NAT Gateways or NAT Instances.

In real time, NAT Gateways are highly used than NAT instances as NAT instances are an individual EC2 instances, and NAT Gateways are highly available across multiple availability zones, and they are not on a single EC2 instance.

**Important points related to NAT Gateways:**

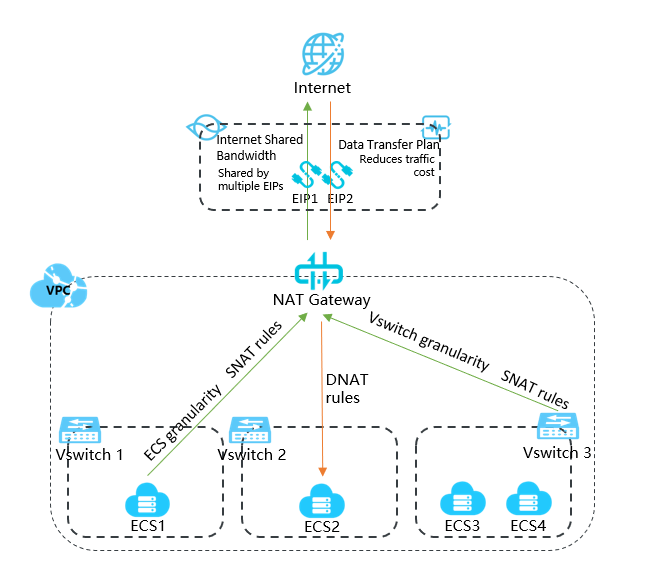
It is redundant inside the availability zone.

It is preferred by an enterprise.

It starts at 5Gbps and scales up to 45 Gbps.

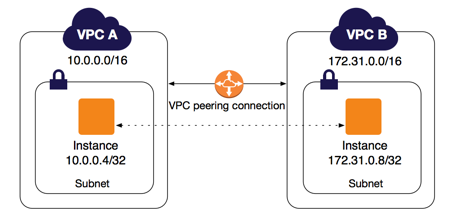
It is not configured with the security groups.

In NAT Gateways, there is no need to disable the source/destination checks.



What is Amazon VPC?

Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.



**NOTEPAD FILE:-**

## Write a Infrastructure as code using terraform, which automatically create a VPC.

**provider "aws" {**

**region = "ap-south-1"**

**profile = "shailly"**

**}**

**resource "aws\_vpc" "shaillyvpc1" {**

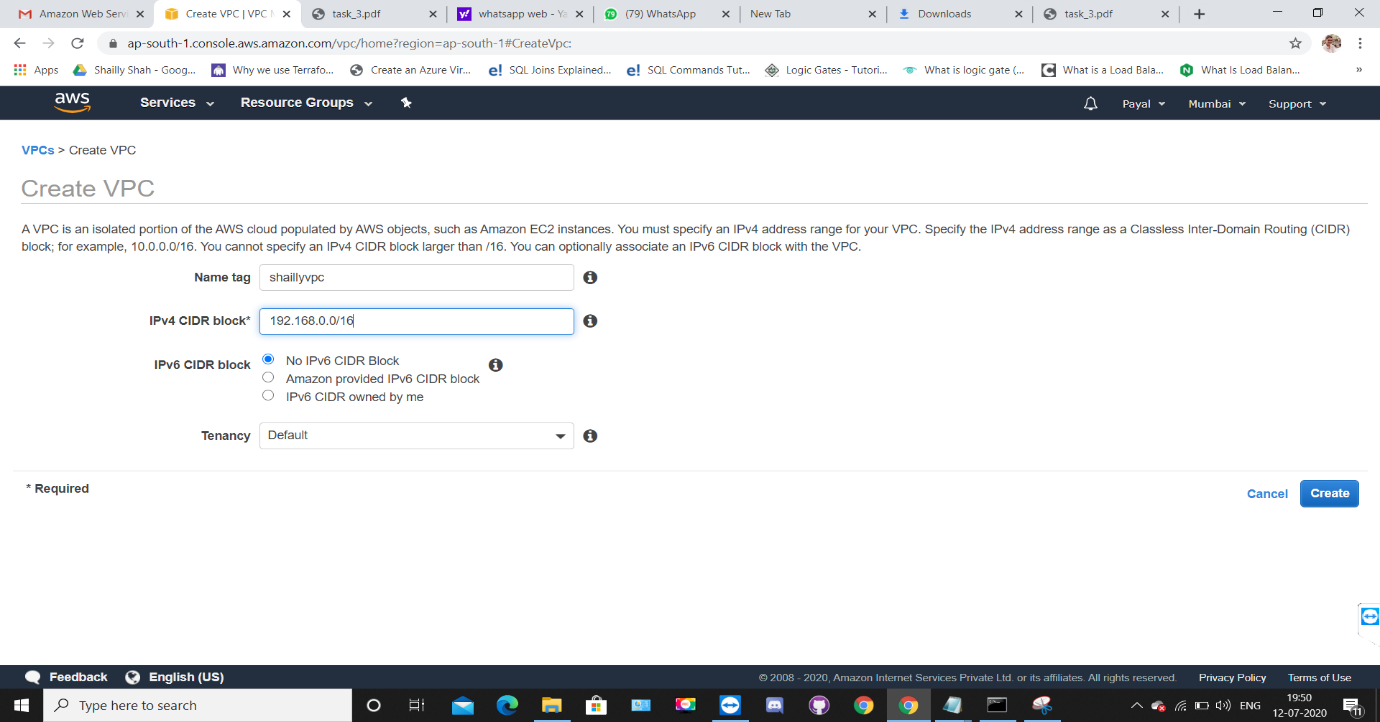
**cidr\_block = "192.168.0.0/16"**

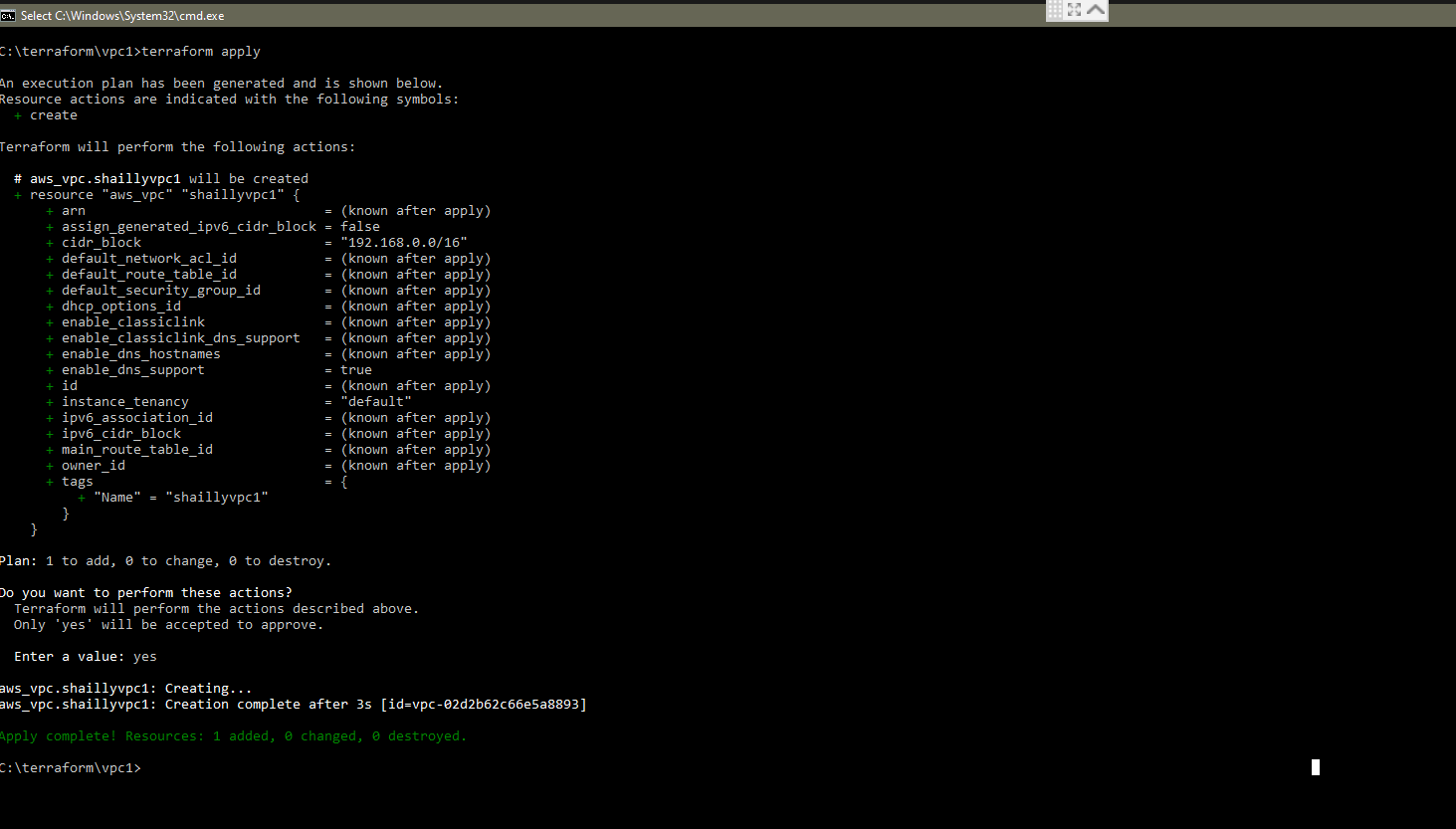
**instance\_tenancy = "default"**

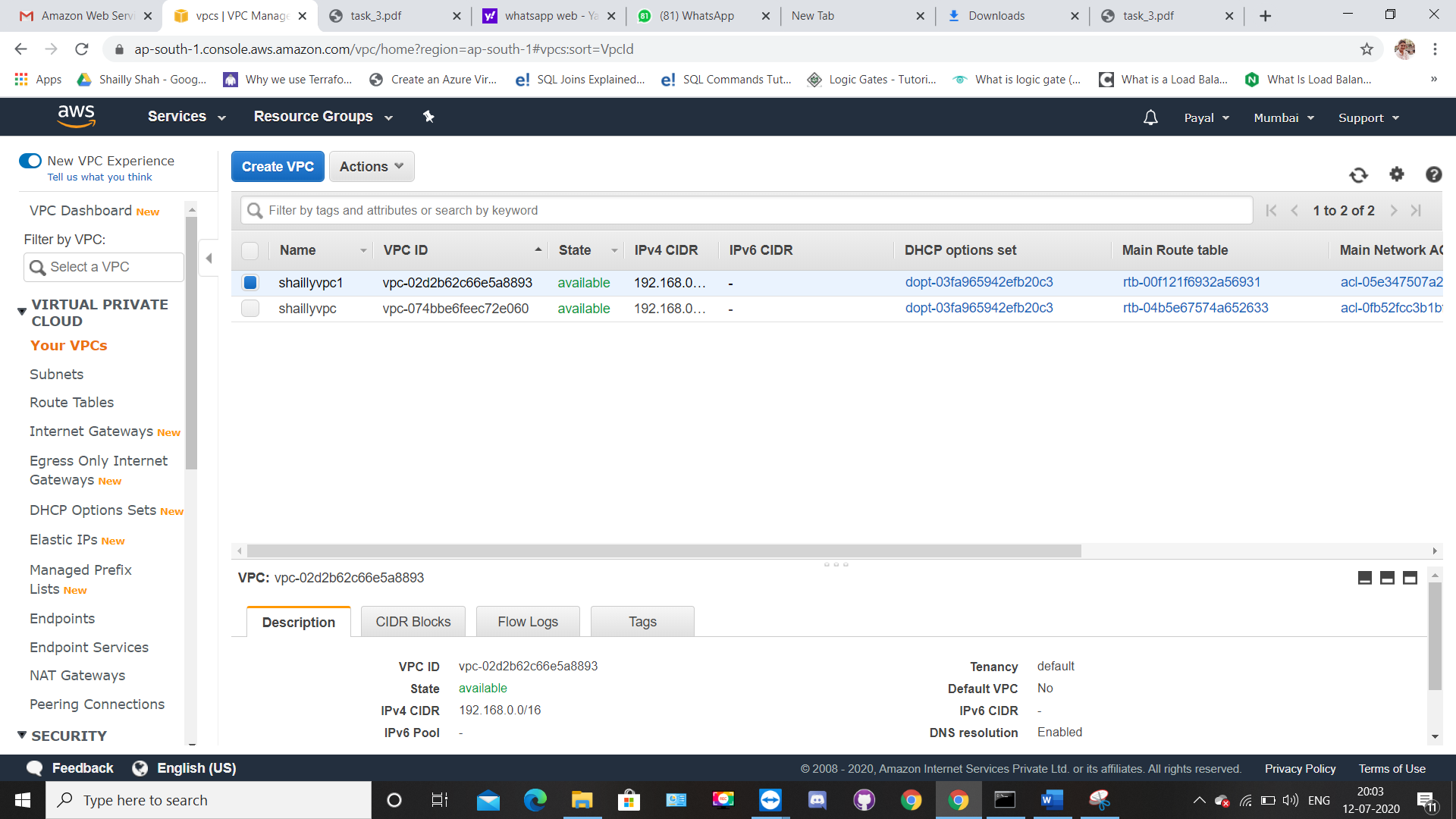
**tags = {**

**Name = "shaillyvpc1"**

**}**

**}**

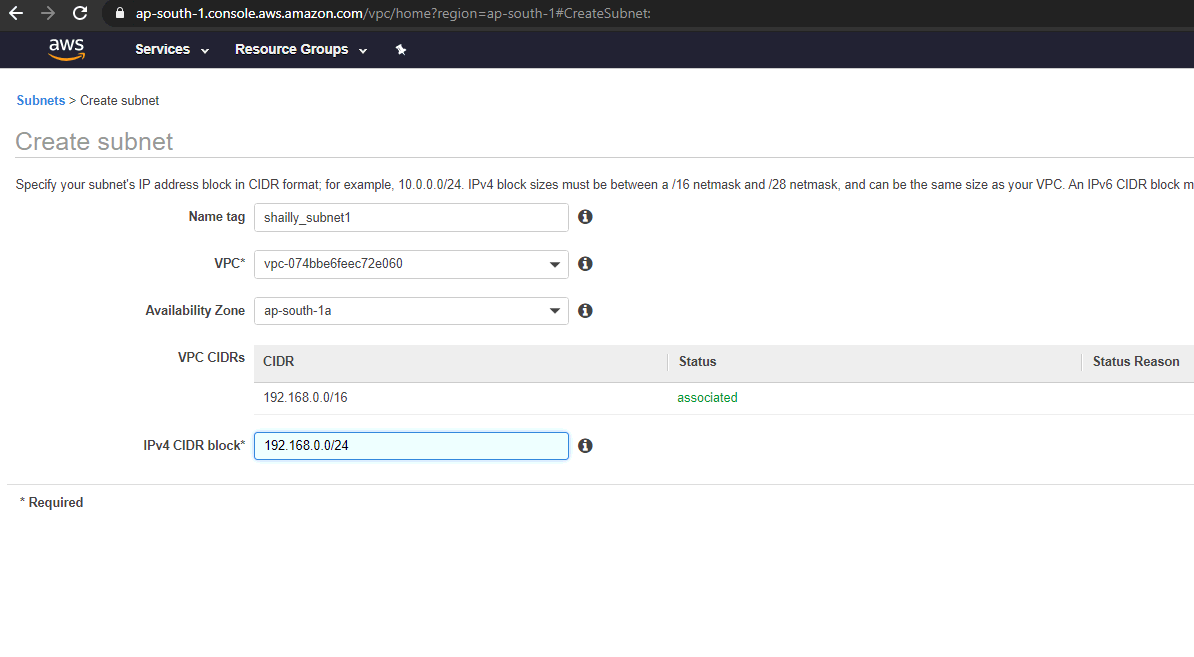


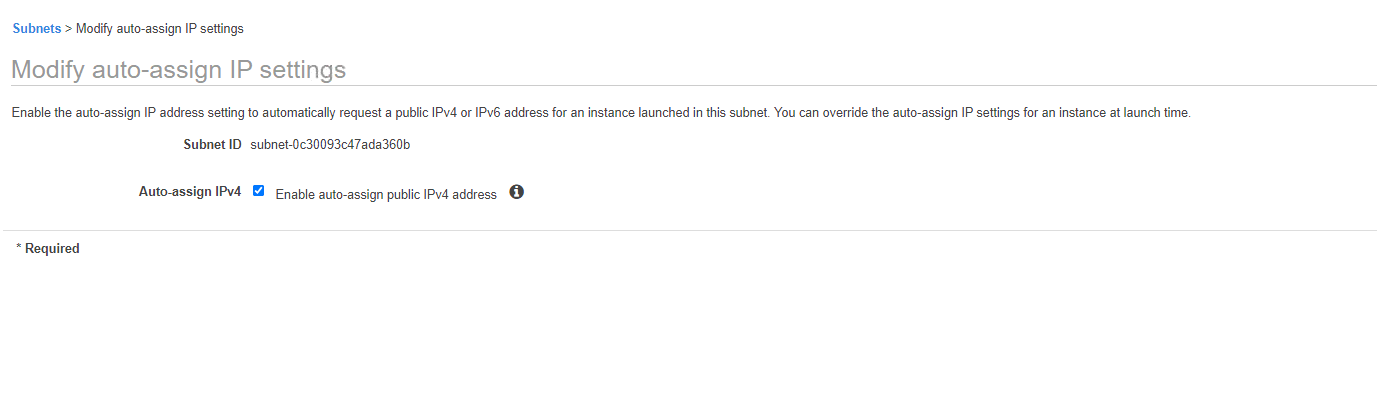


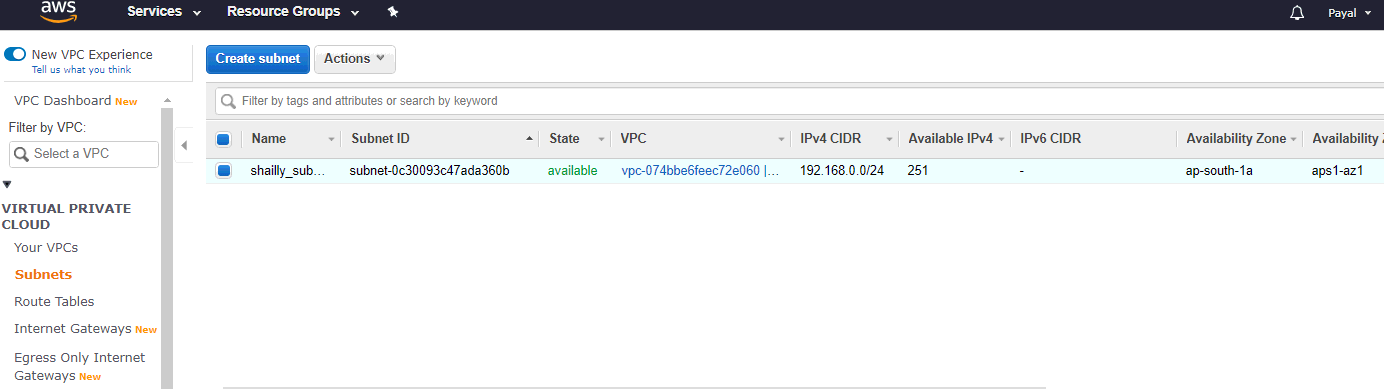
## In that VPC we have to create 2 subnets:

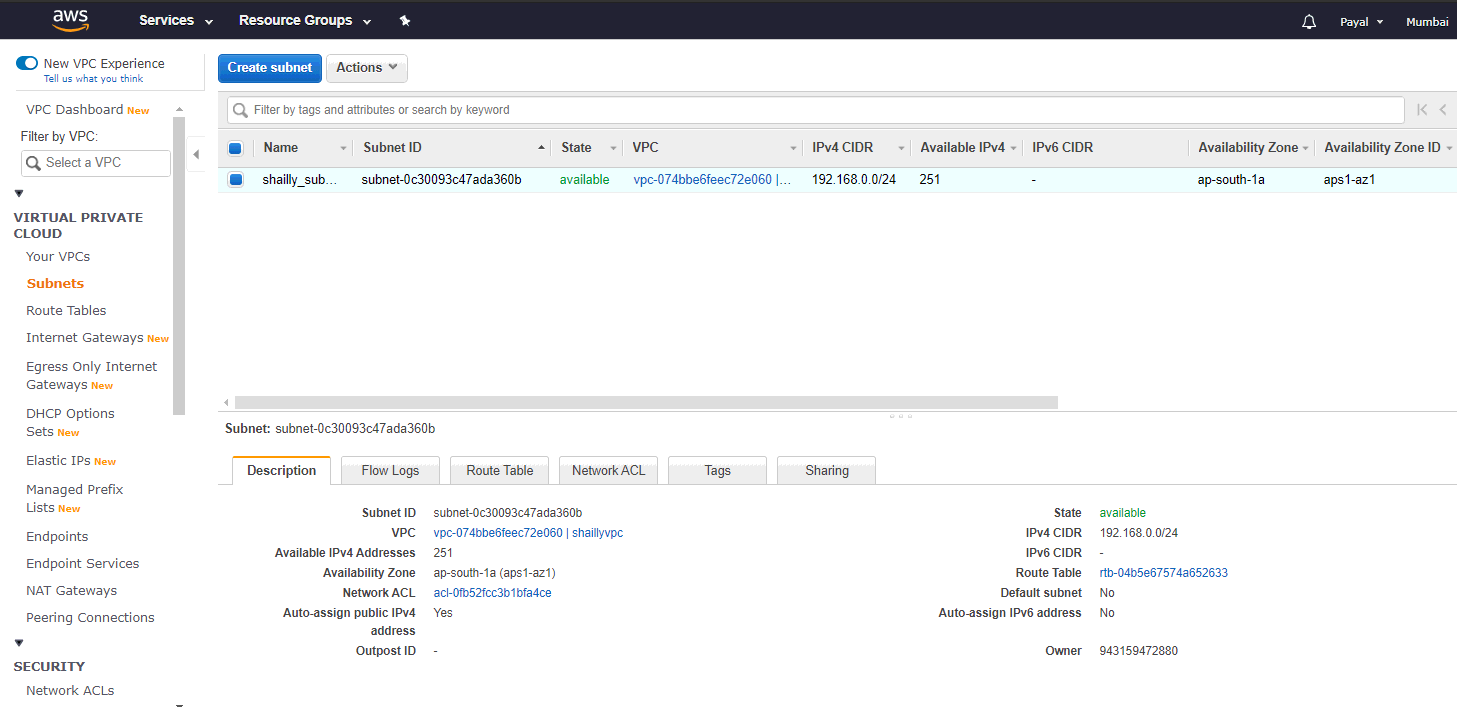
a) public subnet [ Accessible for Public World! ]

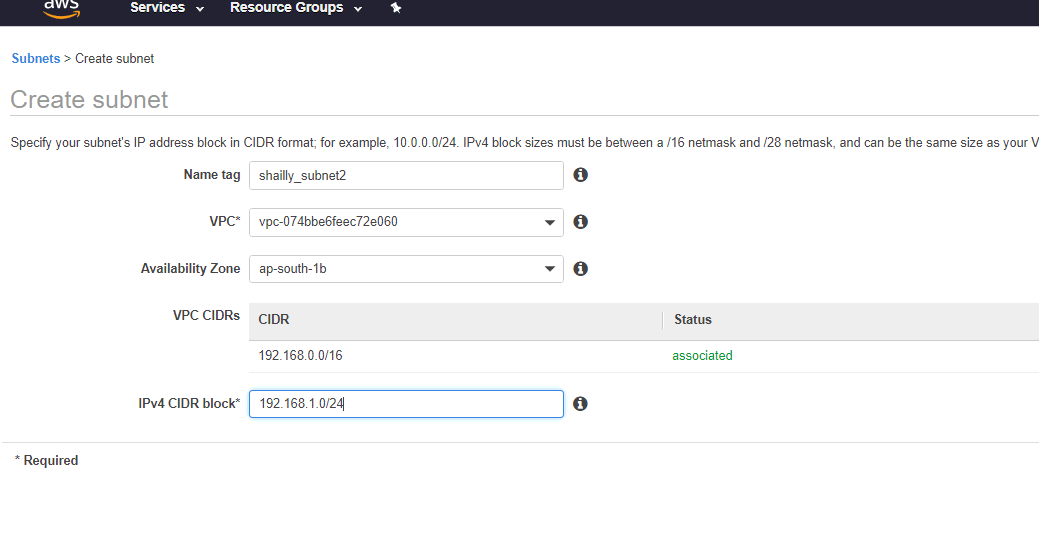
b) private subnet [ Restricted for Public World! ]

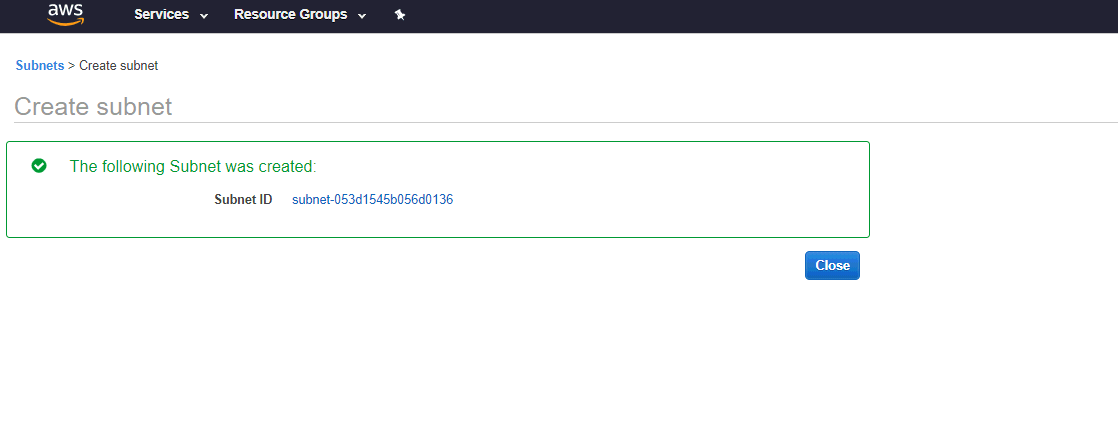


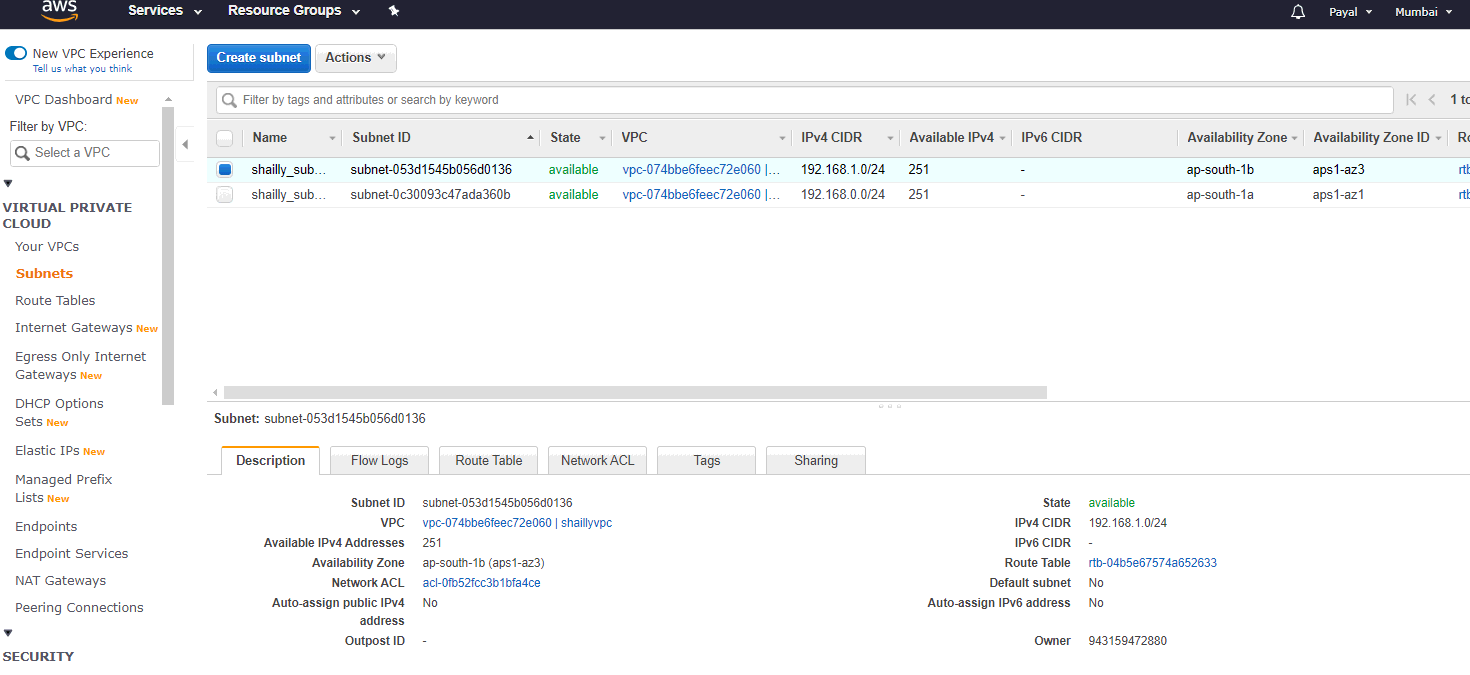












**## using terraform**

**resource "aws\_subnet" "shaillyvpc1\_subnet-1a" {**

**vpc\_id = "${aws\_vpc.shaillyvpc1.id}"**

**cidr\_block = "192.168.0.0/24"**

**availability\_zone = "ap-south-1a"**

**map\_public\_ip\_on\_launch = true**

**}**

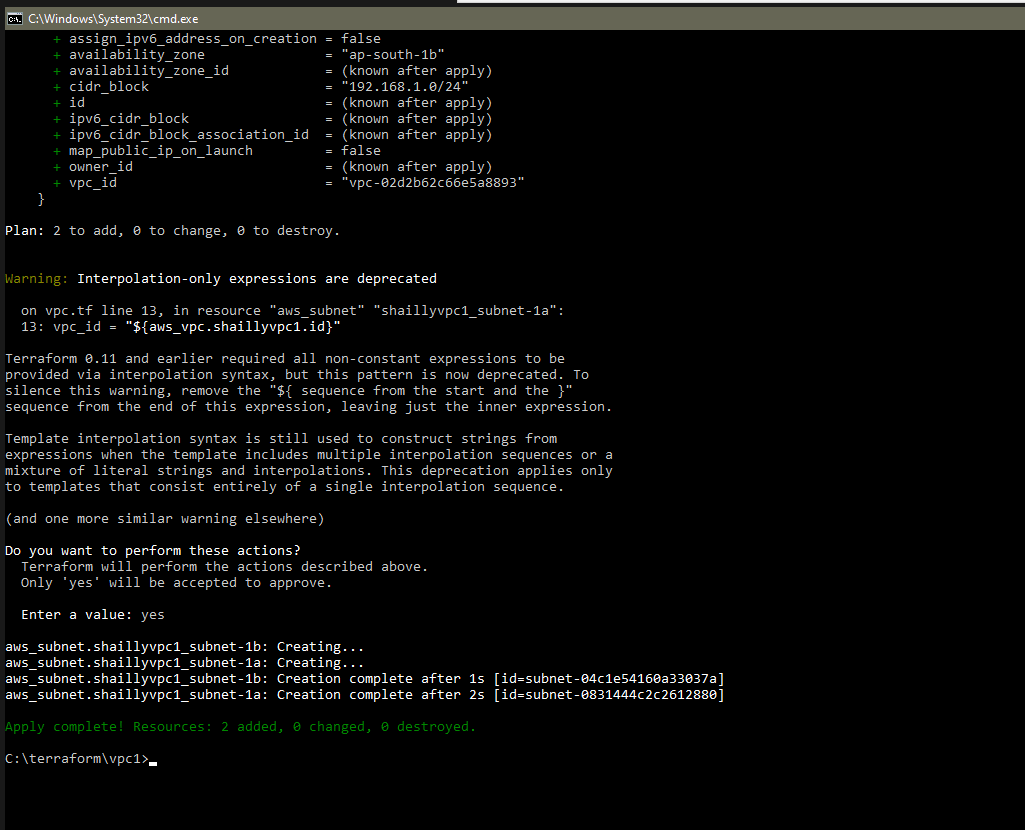
**resource "aws\_subnet" "shaillyvpc1\_subnet-1b" {**

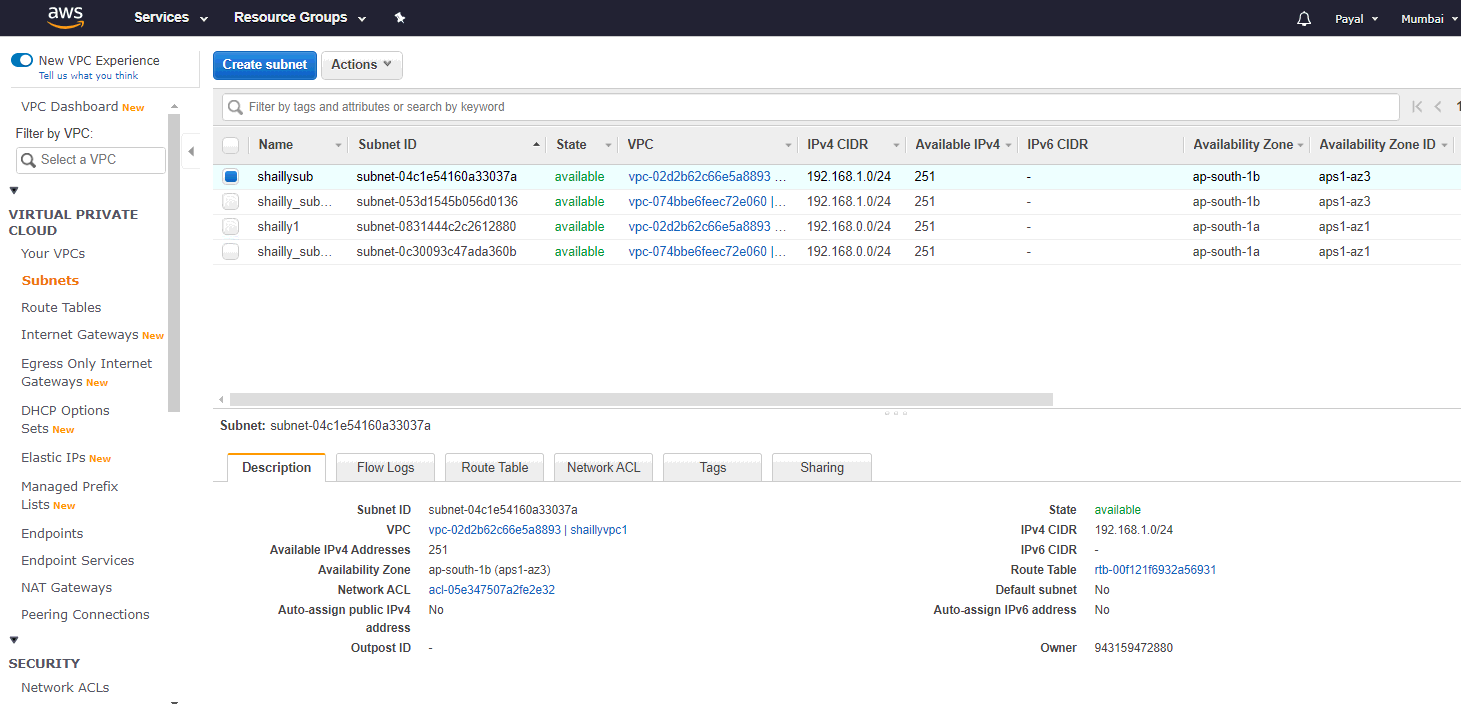
**vpc\_id = "${aws\_vpc.shaillyvpc1.id}"**

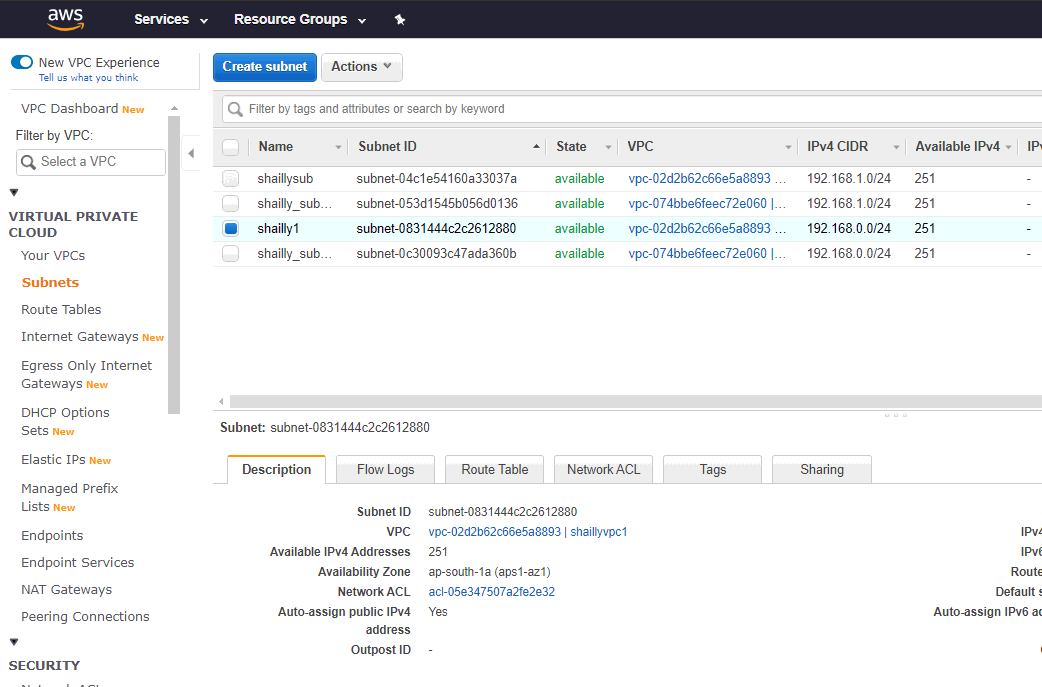
**cidr\_block = "192.168.1.0/24"**

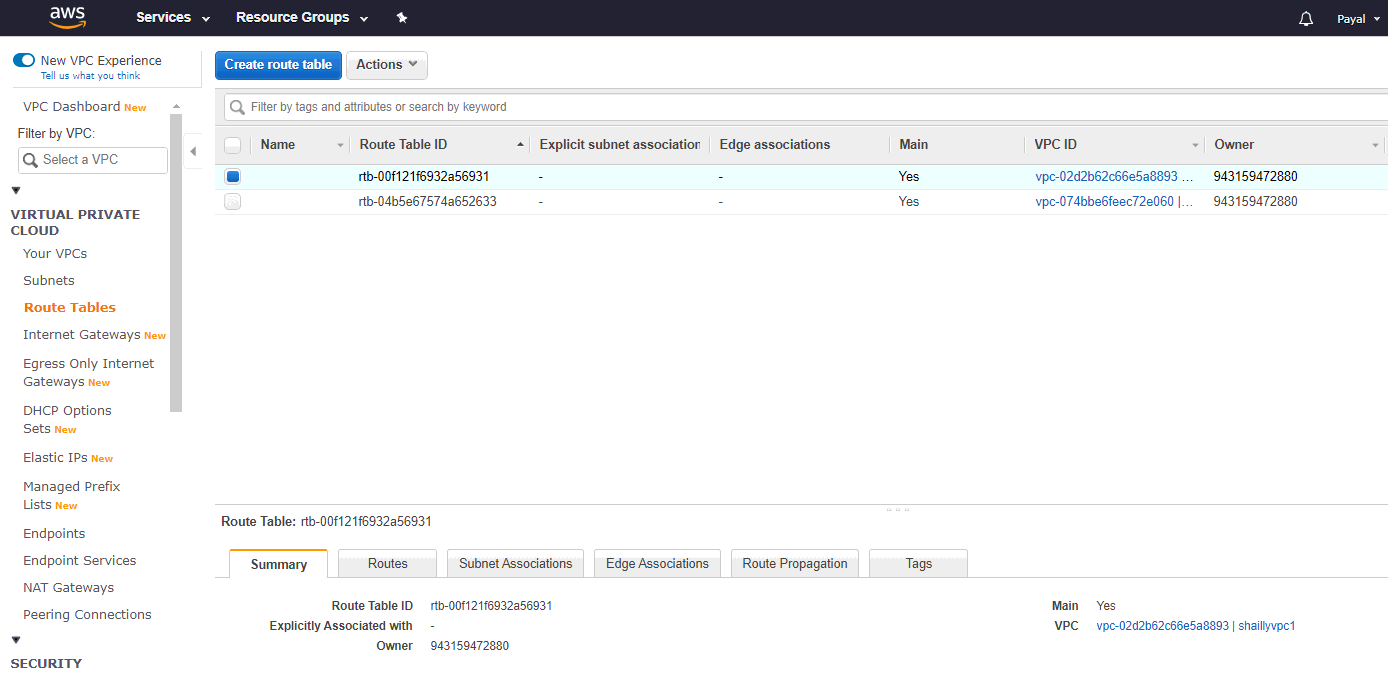
**availability\_zone = "ap-south-1b"**

**}**









## Create a public facing internet gateway for connect our VPC/Network to the internet world and attach this gateway to our VPC.

**resource "aws\_internet\_gateway" "shaillyvpc1\_internet\_gateway" {**

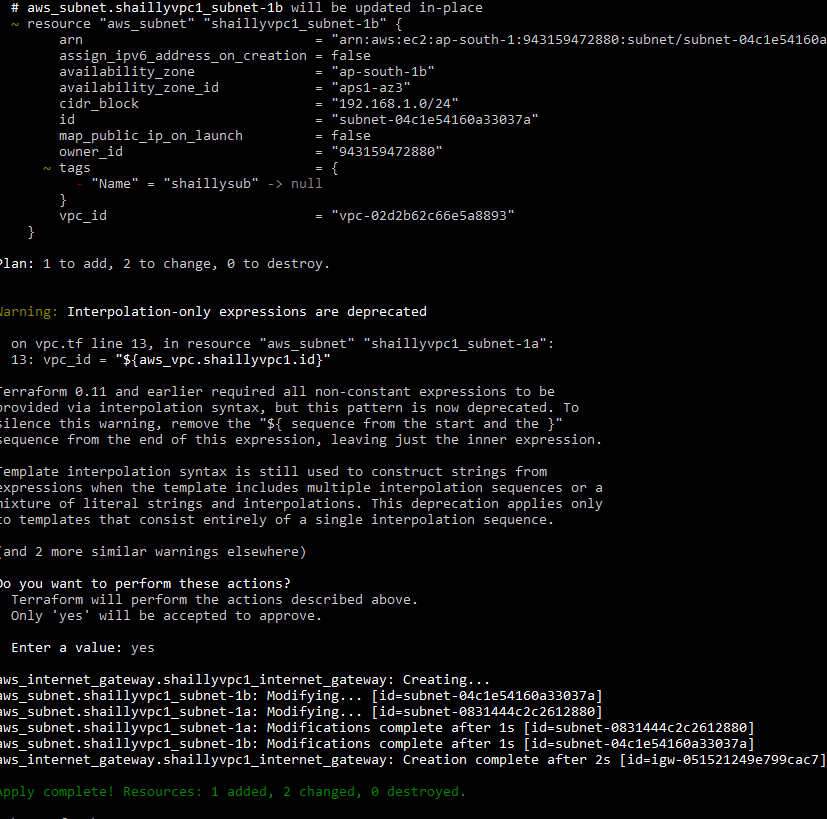
**vpc\_id = "${aws\_vpc.shaillyvpc1.id}"**

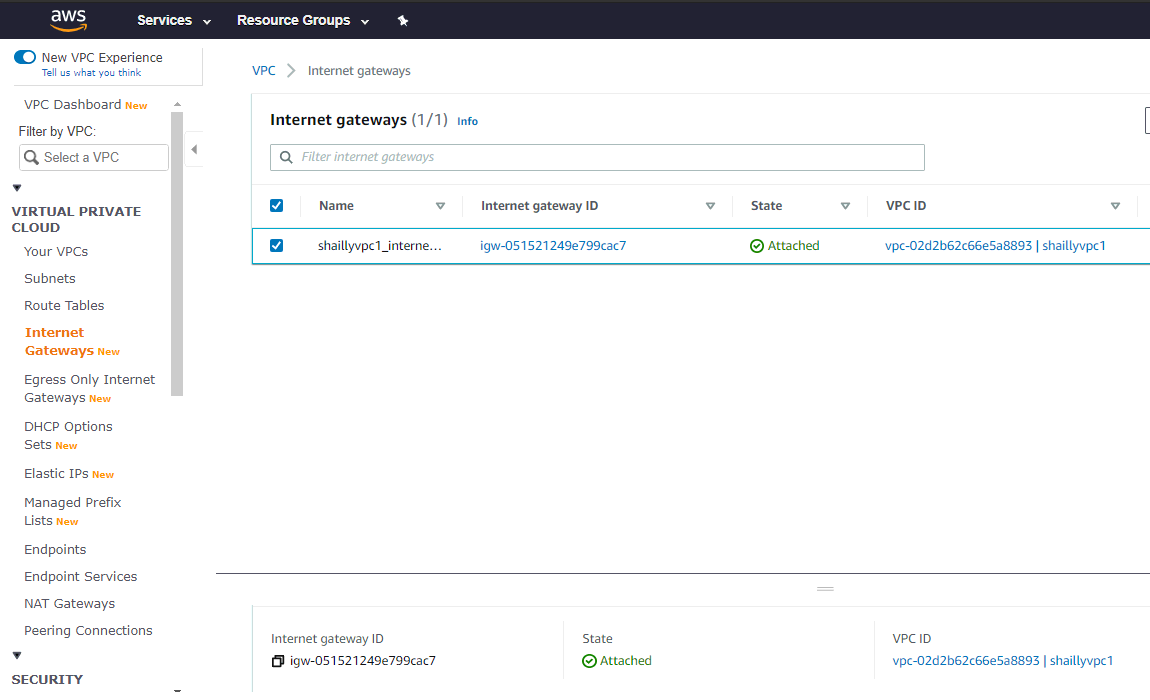
**tags = {**

**Name = "shaillyvpc1\_internet\_gateway"**

**}**

**}**





## Create a routing table for Internet gateway so that instance can connect to outside world, update and associate it with public subnet.

# creating a route-table

# associating route-table with the internet gateway

**resource "aws\_route\_table" "shaillyvpc1\_route\_table" {**

**vpc\_id = "${aws\_vpc.shaillyvpc1.id}"**

**route {**

**cidr\_block = "0.0.0.0/0"**

**gateway\_id = "${aws\_internet\_gateway.shaillyvpc1\_internet\_gateway.id}"**

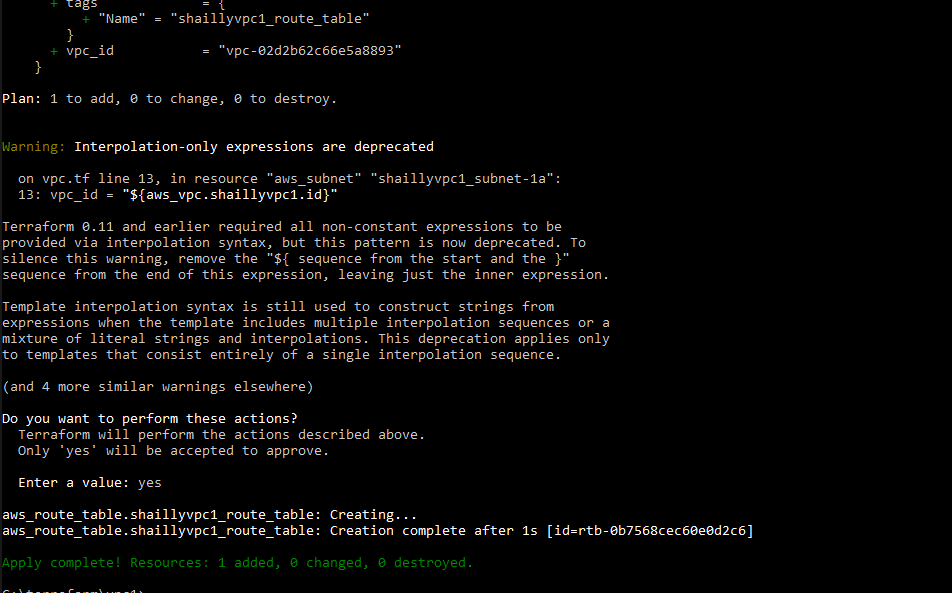
**}**

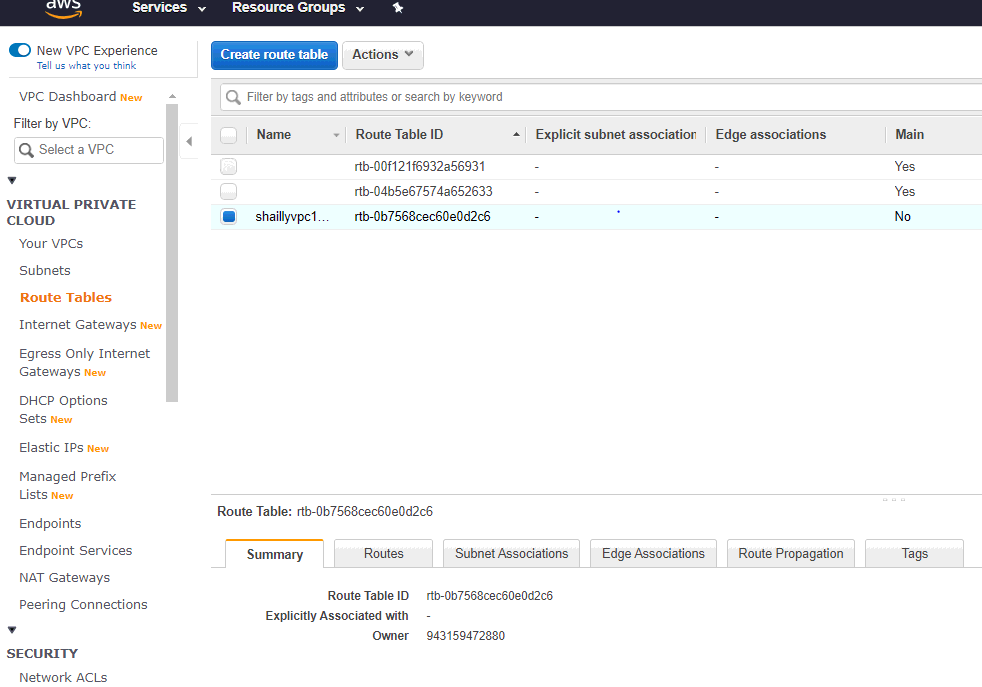
**tags = {**

**Name = "shaillyvpc1\_route\_table"**

**}**

**}**





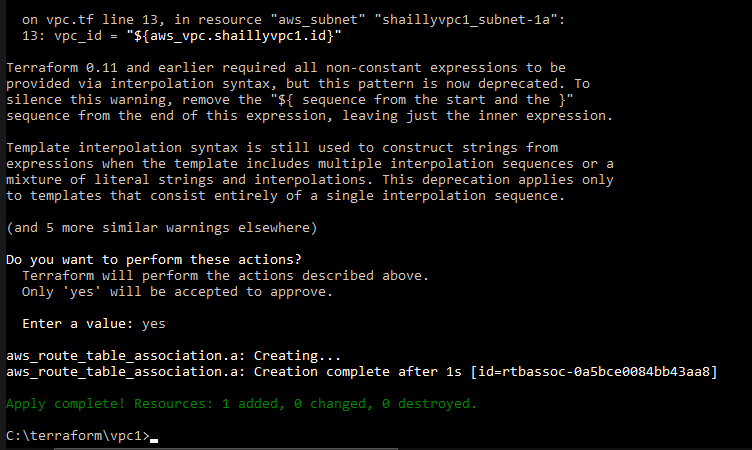
**# associating route table with subnet**

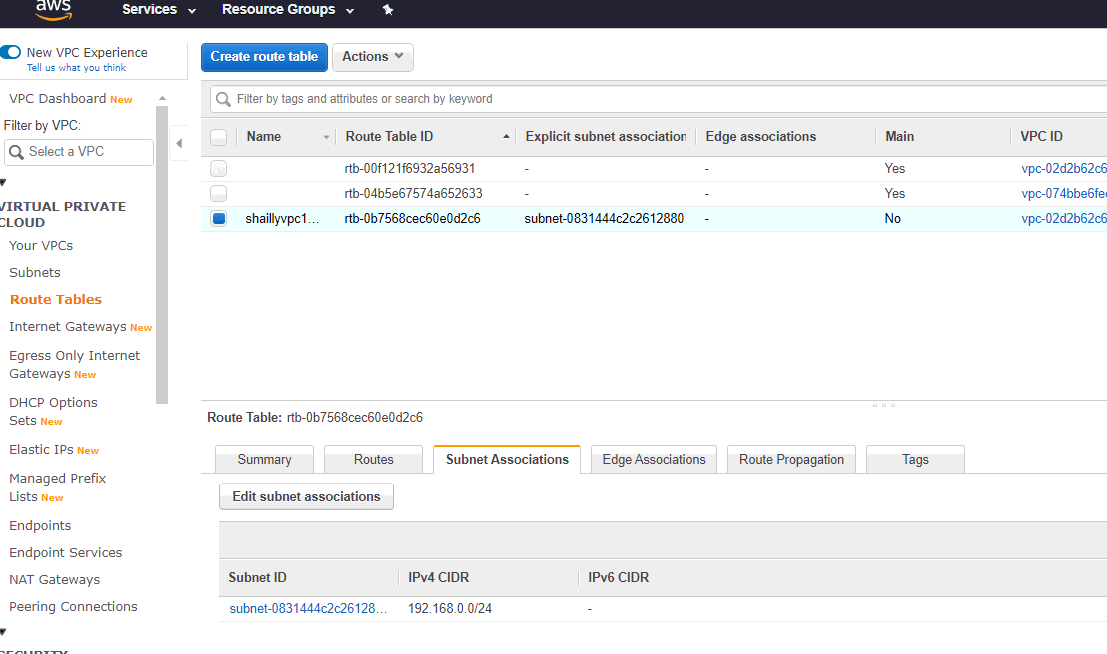
**resource "aws\_route\_table\_association" "a" {**

**subnet\_id = aws\_subnet.shaillyvpc1\_subnet-1a.id**

**route\_table\_id = "${aws\_route\_table.shaillyvpc1\_route\_table.id}"**

**}**





**# creating the security group**

**# shaillyweb**

**resource "aws\_security\_group" "shaillyweb" {**

**name = "shaillyweb"**

**description = "Allow ssh http and icmp"**

**vpc\_id = "${aws\_vpc.shaillyvpc1.id}"**

**ingress {**

**description = "http"**

**from\_port = 80**

**to\_port = 80**

**protocol = "tcp"**

**cidr\_blocks = ["0.0.0.0/0"]**

**}**

**ingress {**

**description = "ssh"**

**from\_port = 22**

**to\_port = 22**

**protocol = "tcp"**

**cidr\_blocks = ["0.0.0.0/0"]**

**}**

**ingress {**

**description = "ICMP-IPv4"**

**from\_port = 0**

**to\_port = 0**

**protocol = "-1"**

**cidr\_blocks = ["0.0.0.0/0"]**

**}**

**egress {**

**from\_port = 0**

**to\_port = 0**

**protocol = "-1"**

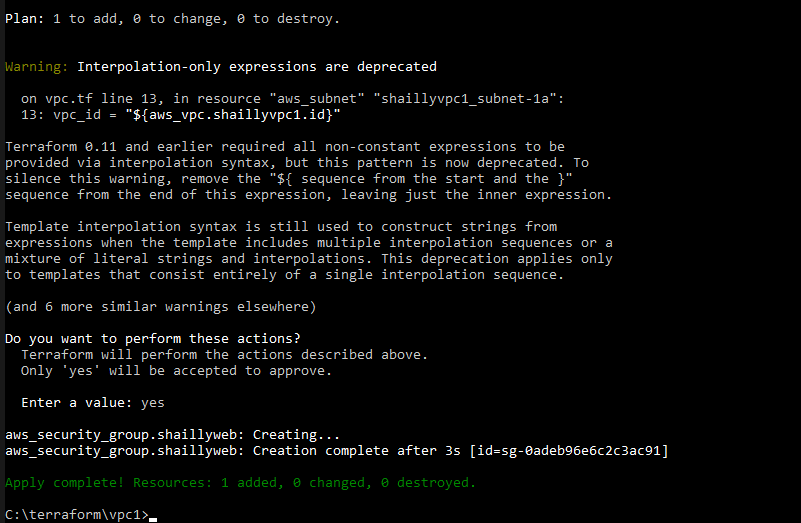
**cidr\_blocks = ["0.0.0.0/0"]**

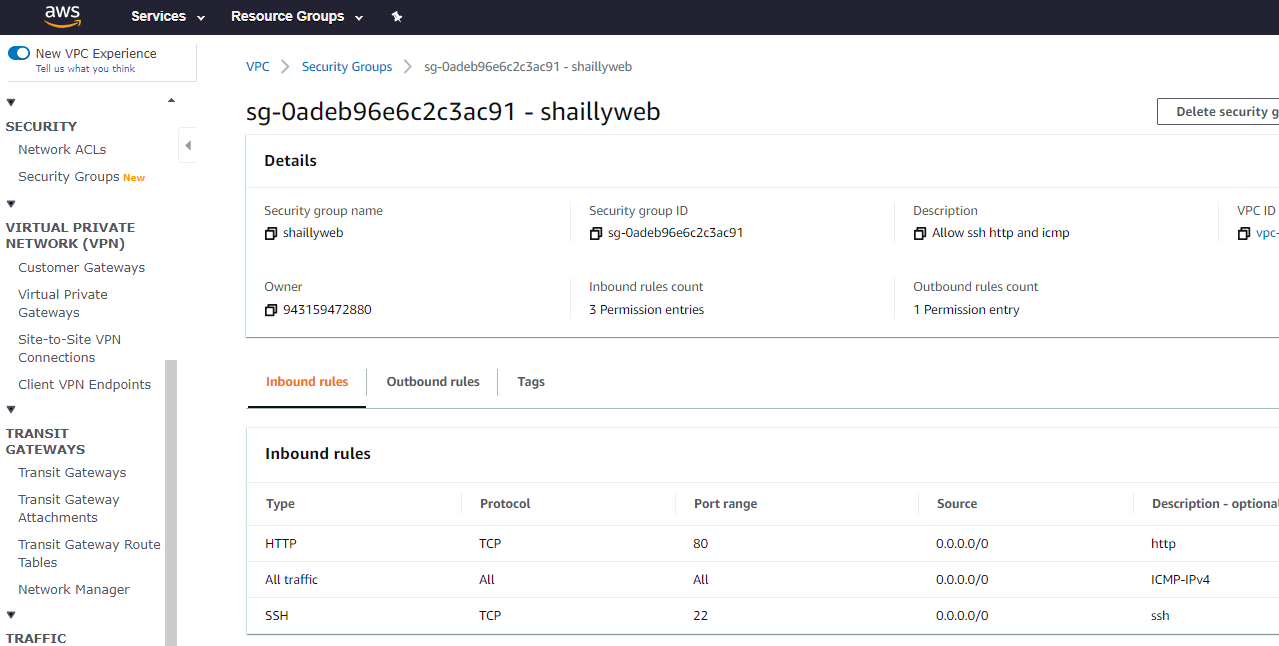
**}**

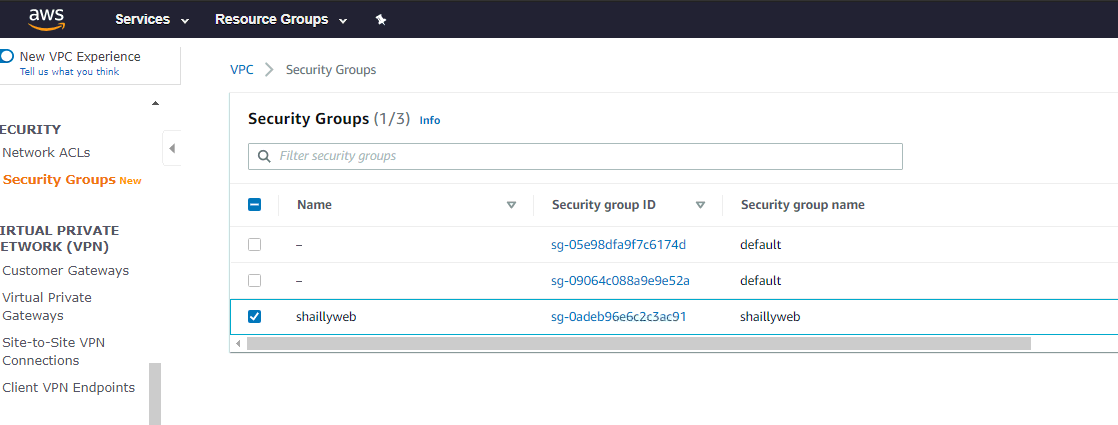
**tags = {**

**Name = "shaillyweb"**

**}}**







**# creating a subnet group with MYSQL protocol and give the value of security id (shaillyweb)**

**# shaillysql**

**resource "aws\_security\_group" "mysql" {**

**name = "shaillysql"**

**description = "Allow sql"**

**vpc\_id = "${aws\_vpc.shaillyvpc1.id}"**

**ingress {**

**description = "MYSQL"**

**security\_groups=[ "${aws\_security\_group.shaillyweb.id}" ]**

**from\_port = 3306**

**to\_port = 3306**

**protocol = "tcp"**

**}**

**egress {**

**from\_port = 0**

**to\_port = 0**

**protocol = "-1"**

**cidr\_blocks = ["0.0.0.0/0"]**

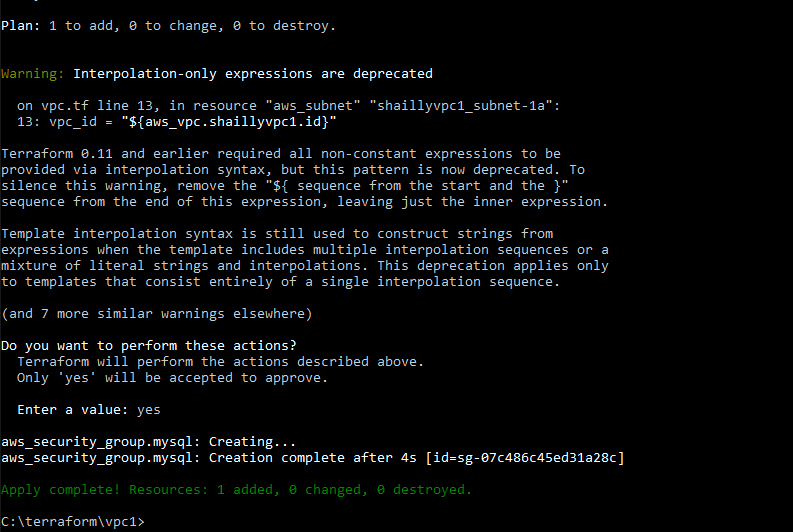
**}**

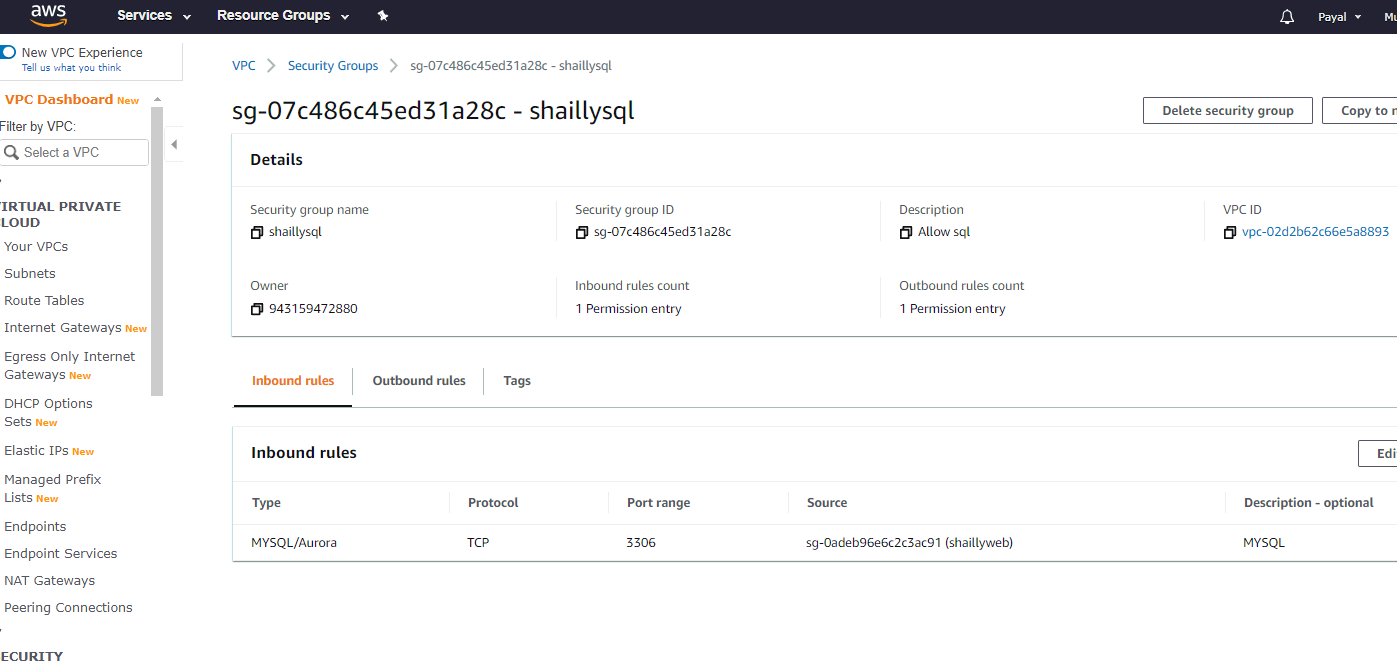
**tags = {**

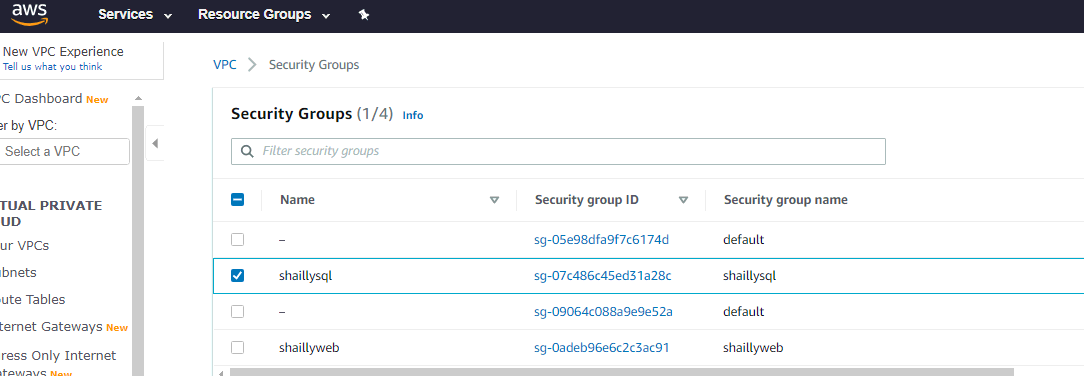
**Name = "shaillysql"**

**}**

**}**







**# creating a security group with ssh protocol**

**# bastionhost**

**resource "aws\_security\_group" "shahbastion" {**

**name = "shahbastion"**

**description = "Allow ssh for bastion"**

**vpc\_id = "${aws\_vpc.shaillyvpc1.id}"**

**ingress {**

**description = "ssh"**

**from\_port = 22**

**to\_port = 22**

**protocol = "tcp"**

**cidr\_blocks = ["0.0.0.0/0"]**

**}**

**egress {**

**from\_port = 0**

**to\_port = 0**

**protocol = "-1"**

**cidr\_blocks = ["0.0.0.0/0"]**

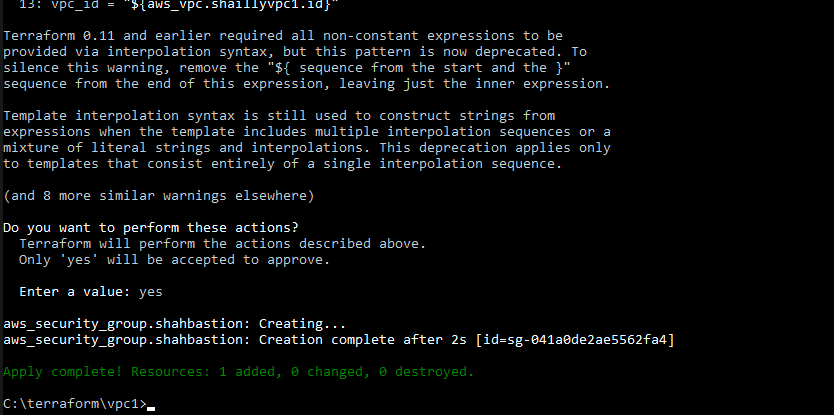
**}**

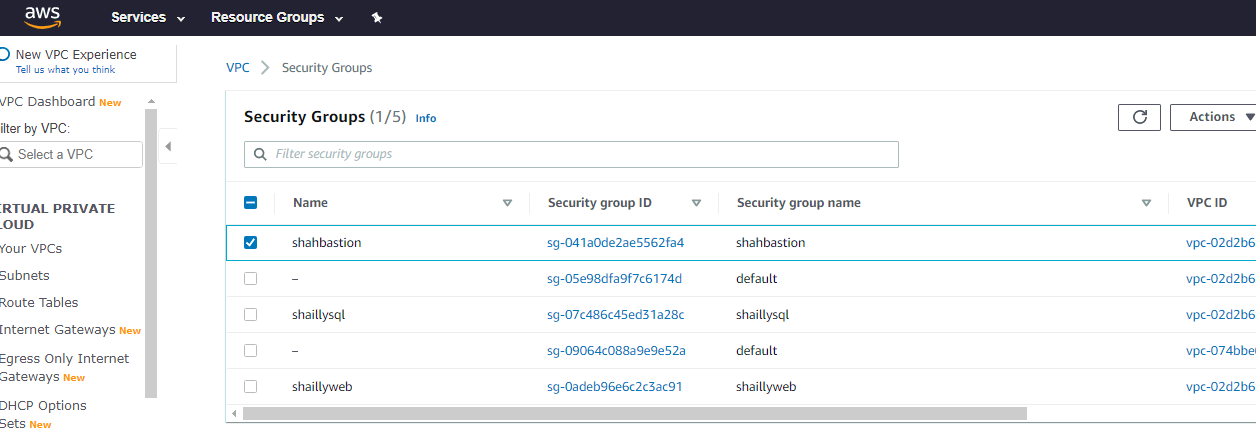
**tags = {**

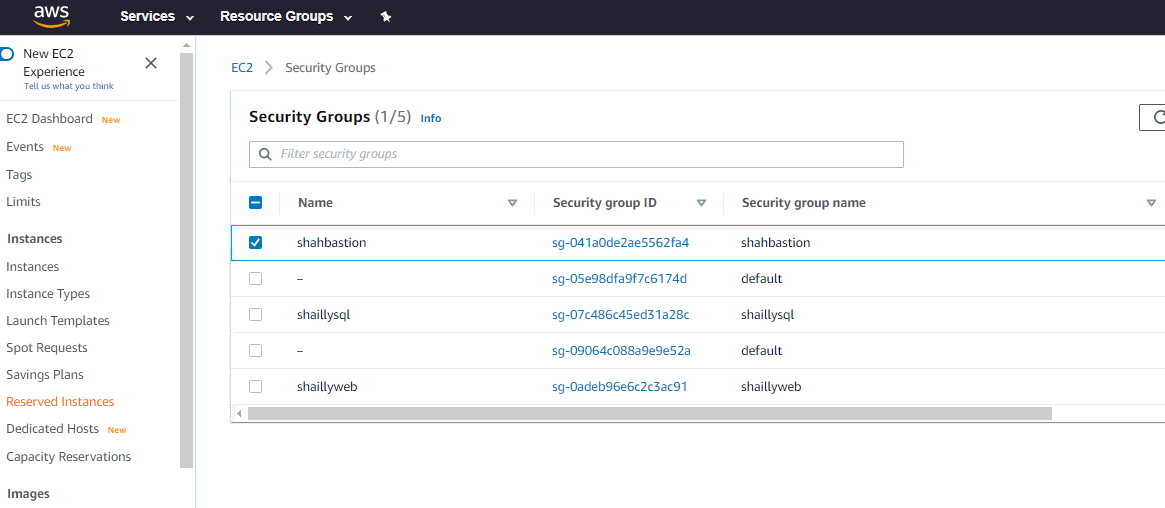
**Name = "shahbastion"**

**}**

**}**







**# creating a subnet group with ssh protocol**

**# shahsqlallow**

**resource "aws\_security\_group" "shahsqlallow" {**

**name = "shahsqlallow"**

**description = "ssh allow to the mysql"**

**vpc\_id = "${aws\_vpc.shaillyvpc1.id}"**

**ingress {**

**description = "ssh"**

**security\_groups=[ "${aws\_security\_group.shahbastion.id}" ]**

**from\_port = 22**

**to\_port = 22**

**protocol = "tcp"**

**}**

**egress {**

**from\_port = 0**

**to\_port = 0**

**protocol = "-1"**

**cidr\_blocks = ["0.0.0.0/0"]**

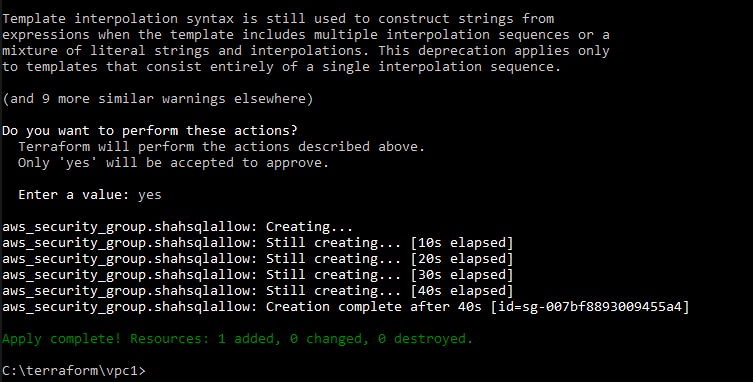
**}**

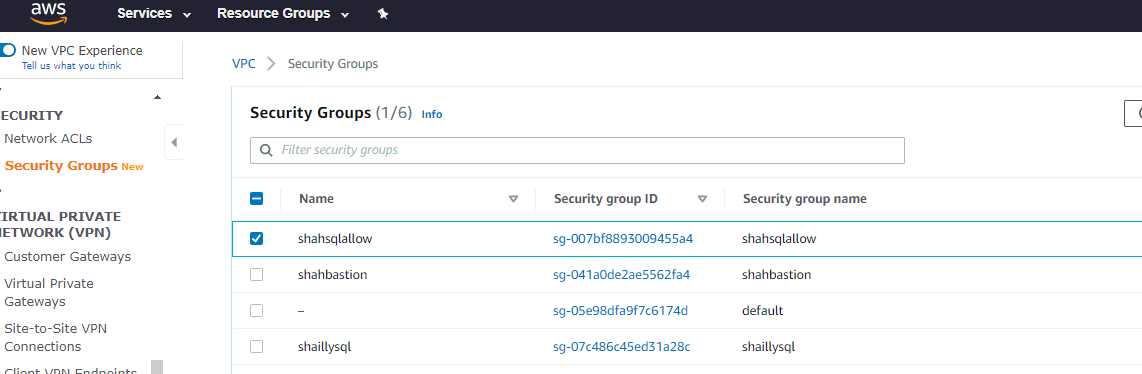
**tags = {**

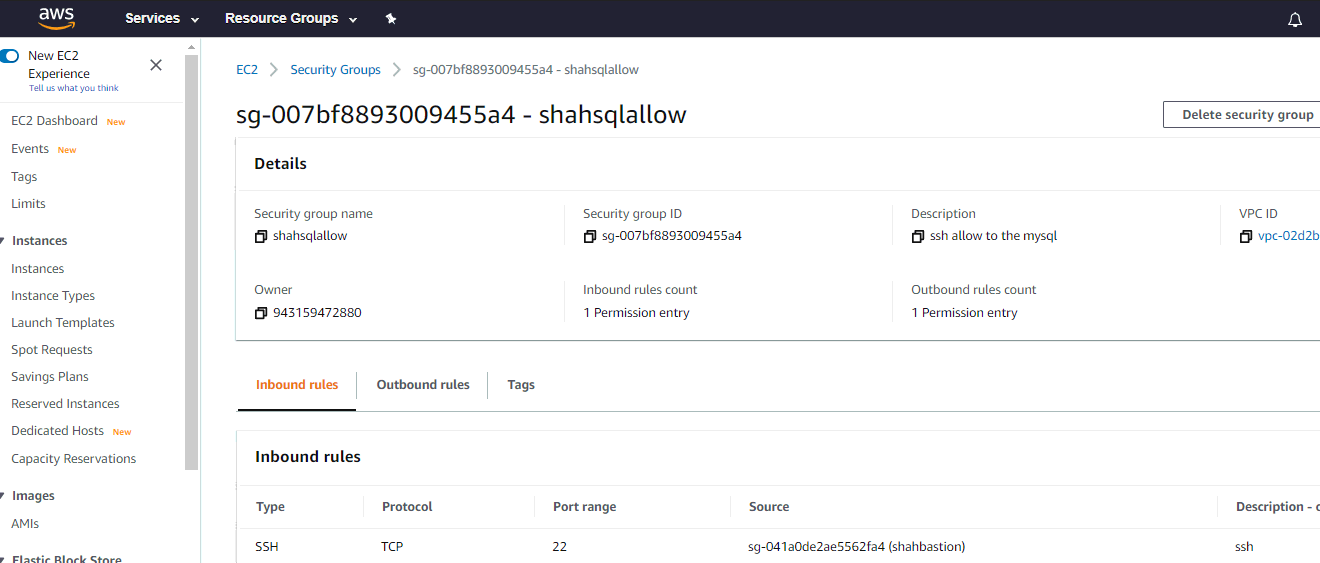
**Name = "shahsqlallow"**

**}**

**}**







##. Create a NAT gateway for connect our VPC/Network to the internet world and attach this gateway to our VPC in the public network/// Update the routing table of the private subnet, so that to access the internet it uses the nat gateway created in the public subnet

**resource "aws\_eip" "shaillyvpc1\_eip" {**

**vpc = true**

**}**

**resource "aws\_nat\_gateway" "shaillyvpc1\_nat\_gateway" {**

**allocation\_id = "${aws\_eip.shaillyvpc1\_eip.id}"**

**subnet\_id = "${aws\_subnet.shaillyvpc1\_subnet-1b.id}"**

**tags = {**

**Name = "shaillyvpc1\_nat\_gateway"**

**}**

**}**

**resource "aws\_route\_table" "shaillyvpc1\_route\_table2" {**

**vpc\_id = "${aws\_vpc.shaillyvpc1.id}"**

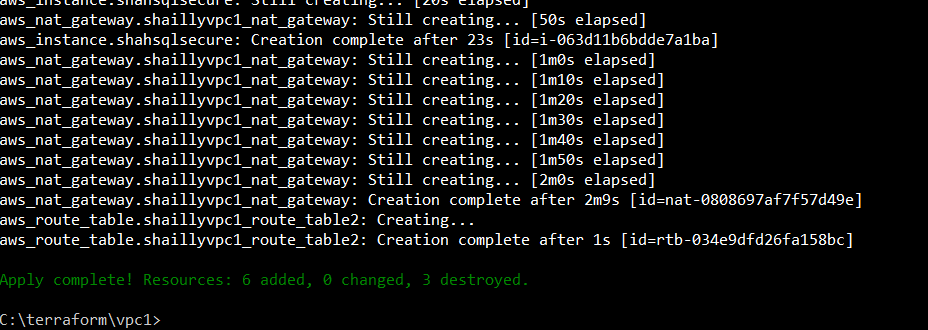
**route {**

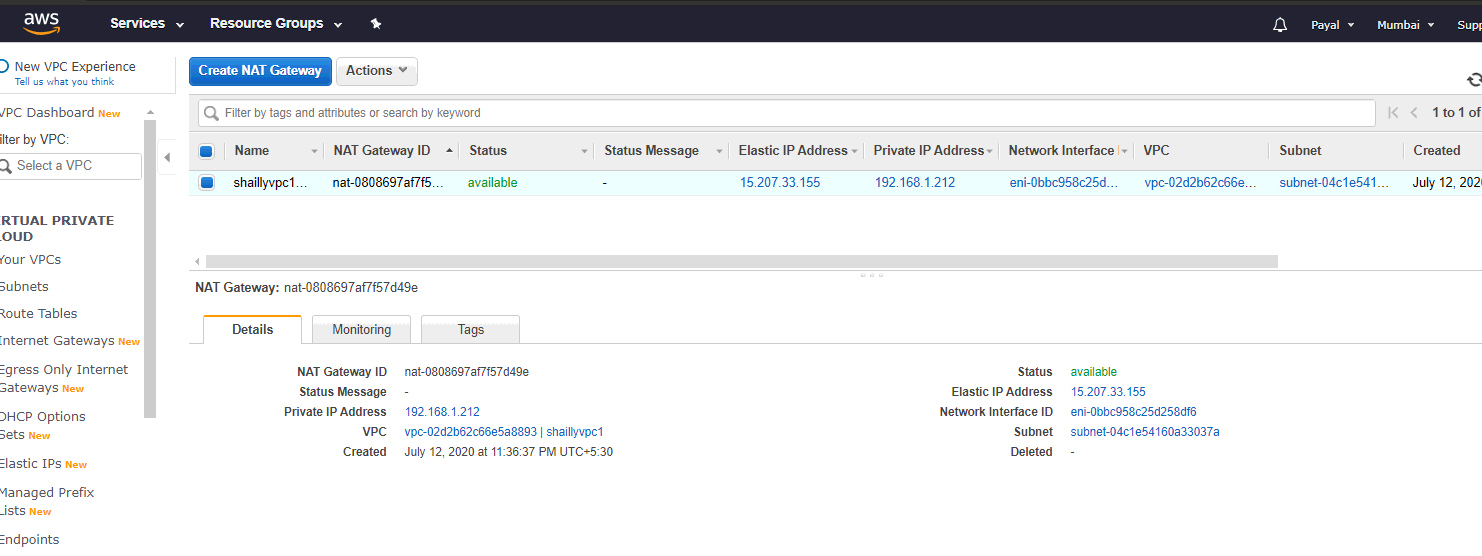
**cidr\_block = "0.0.0.0/0"**

**nat\_gateway\_id = "${aws\_nat\_gateway.shaillyvpc1\_nat\_gateway.id}"**

**}**

**}**





##. Launch an ec2 instance which has Wordpress setup already having the security group allowing port 80 so that our client can connect to our wordpress site.

Also attach the key to instance for further login into it.

**resource "aws\_instance" "shahwordpress" {**

**ami = "ami-0cb39c5da8e2fa515"**

**instance\_type = "t2.micro"**

**key\_name = "shailly.pem"**

**availability\_zone = "ap-south-1a"**

**subnet\_id = "${aws\_subnet.shaillyvpc1\_subnet-1a.id}"**

**security\_groups = [ "${aws\_security\_group.shaillyweb.id}" ]**

**user\_data = <<-EOF**

**#! /bin/bash**

**sudo yum install dnf install php-mysqlnd php-fpm httpd tar curl php-json -y**

**systemctl start httpd**

**systemctl enable httpd**

**curl https://wordpress.org/latest.tar.gz --output wordpress.tar.gz**

**tar xf wordpress.tar.gz**

**cp -r wordpress /var/www/html**

**chown -R apache:apache /var/www/html/wordpress**

**chcon -t httpd\_sys\_rw\_content\_t /var/www/html/wordpress -R**

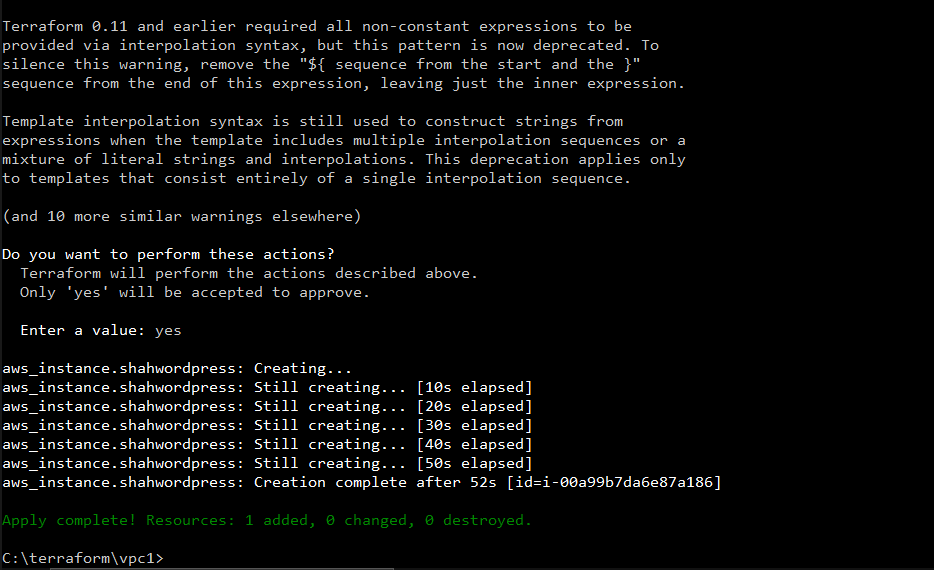
**EOF**

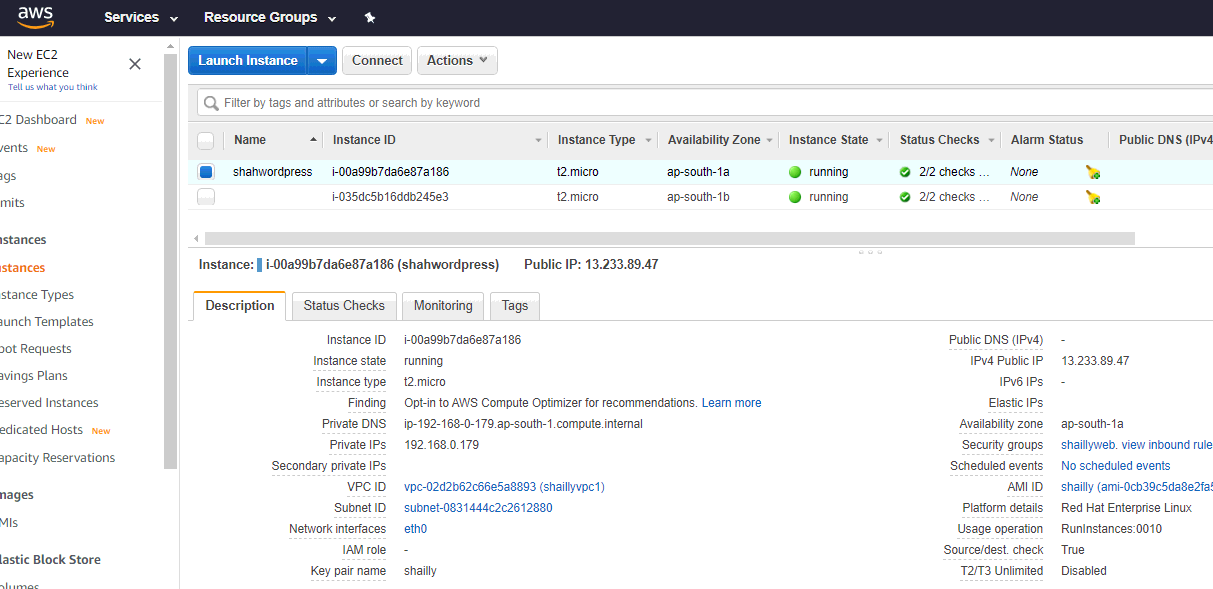
**tags = {**

**Name = "shahwordpress"**

**}**

**}**





##. Launch an ec2 instance which has MYSQL setup already with security group allowing port 3306 in private subnet so that our wordpress vm can connect with the same.

Also attach the key with the same.

**# launching the instance with the rhel image in the region 1b and attaching the security group shaillysql and shaillysqlallow**

**# not enabling the public-ip**

**resource "aws\_instance" "shahsqlsecure" {**

**ami = "ami-02c9b9770f41dc7de"**

**instance\_type = "t2.micro"**

**key\_name = "shailly"**

**availability\_zone = "ap-south-1b"**

**subnet\_id = "${aws\_subnet.shaillyvpc1\_subnet-1b.id}"**

**security\_groups = [ "${aws\_security\_group.mysql.id}" ,**

**"${aws\_security\_group.shahsqlallow.id}"]**

**user\_data = <<-EOF**

**#! /bin/bash**

**sudo yum install @shaillysql -y**

**systemctl start shaillysqld**

**systemctl enable shaillysqld**

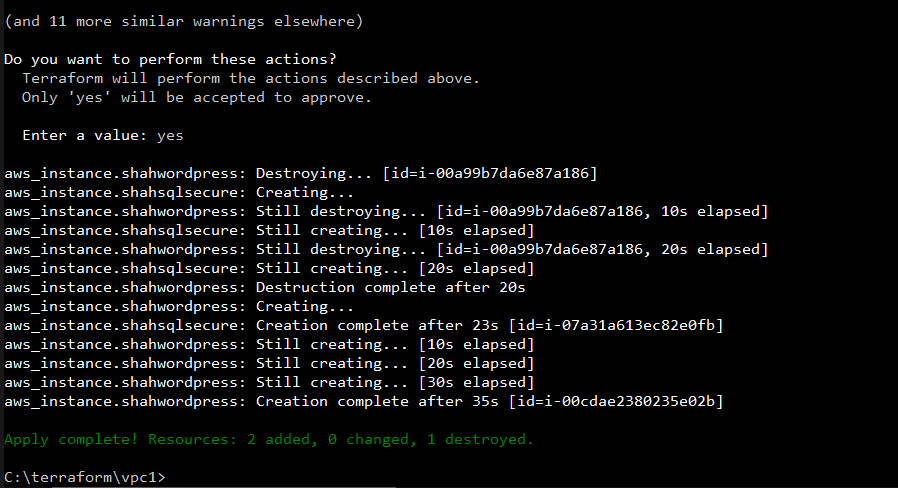
**EOF**

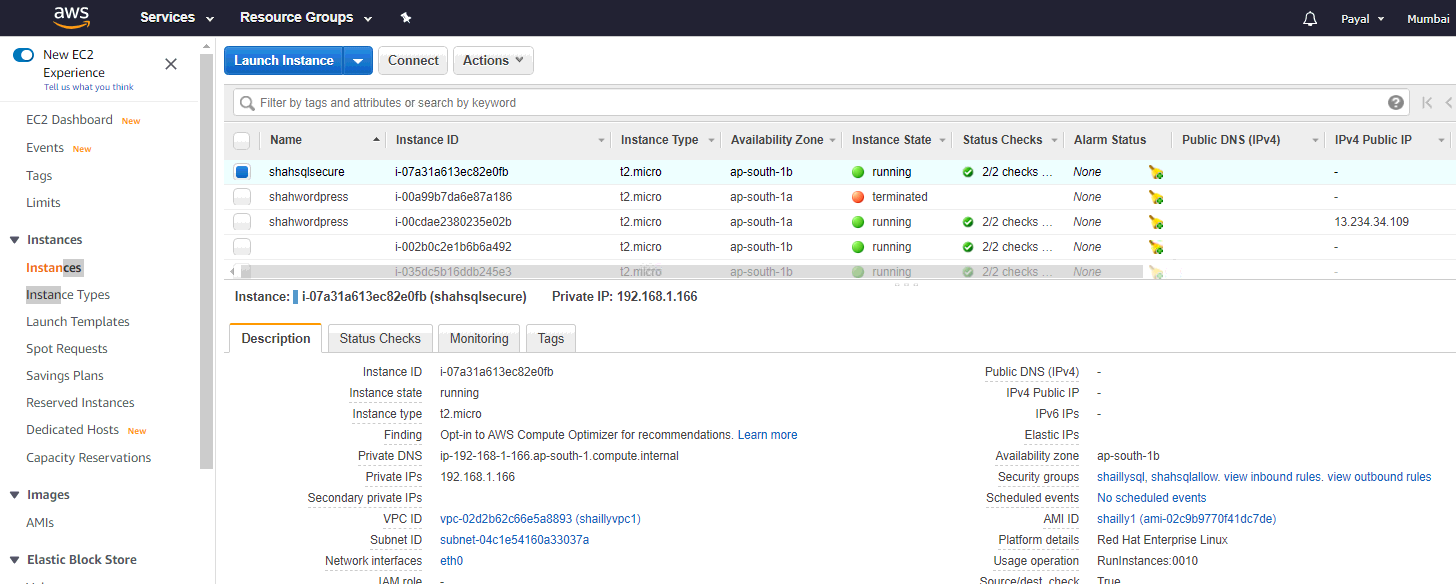
**tags = {**

**Name = "shahsqlsecure"**

**}**

**}**





**# launching the instance with the rhel image in the region 1a and attaching the security group shahbastion**

**# enabling the public-ip**

**resource "aws\_instance" "shahbastion" {**

**ami = "ami-073a8ab1b15e272e5"**

**instance\_type = "t2.micro"**

**key\_name = "shailly"**

**availability\_zone = "ap-south-1a"**

**subnet\_id = "${aws\_subnet.shaillyvpc1\_subnet-1a.id}"**

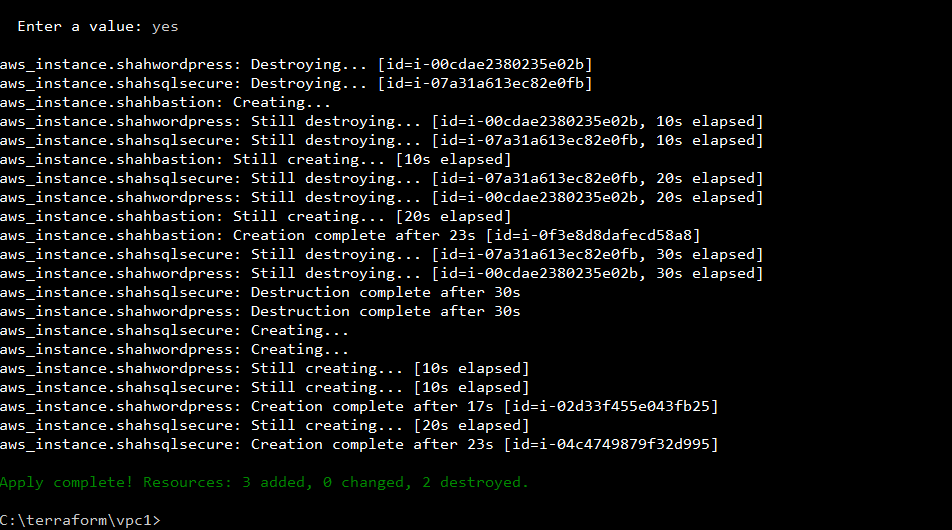
**security\_groups = [ "${aws\_security\_group.shahbastion.id}" ]**

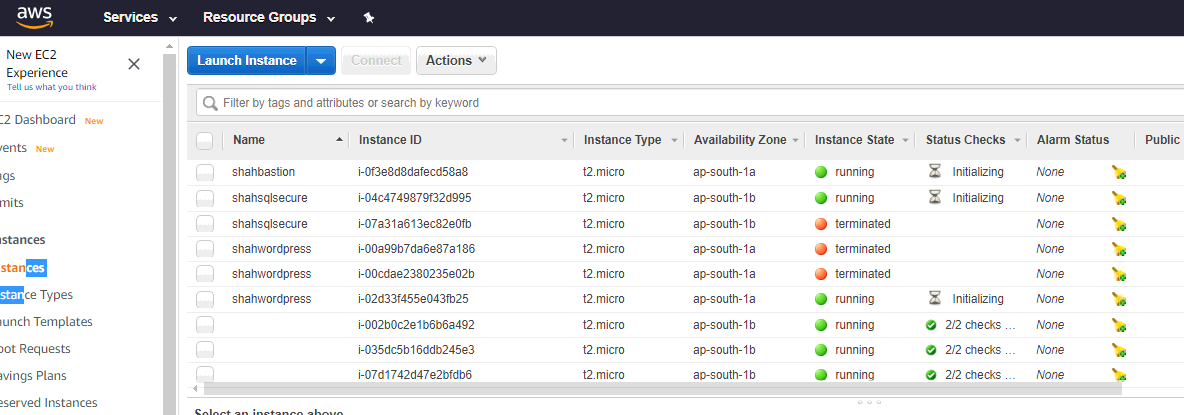
**tags = {**

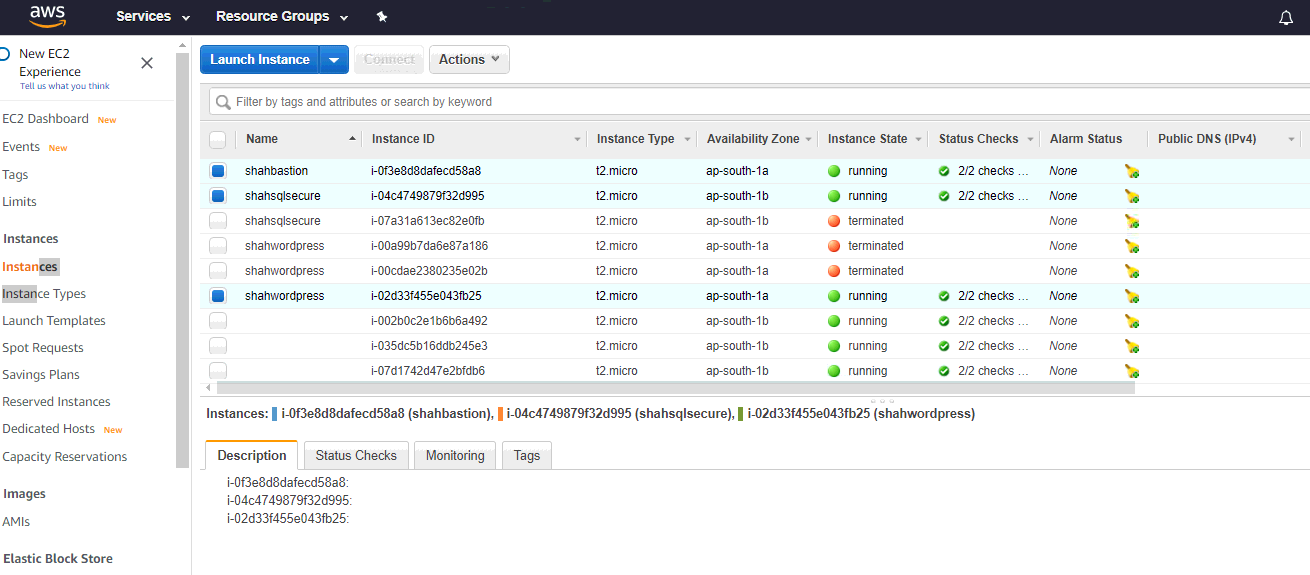
**Name = "shahbastion"**

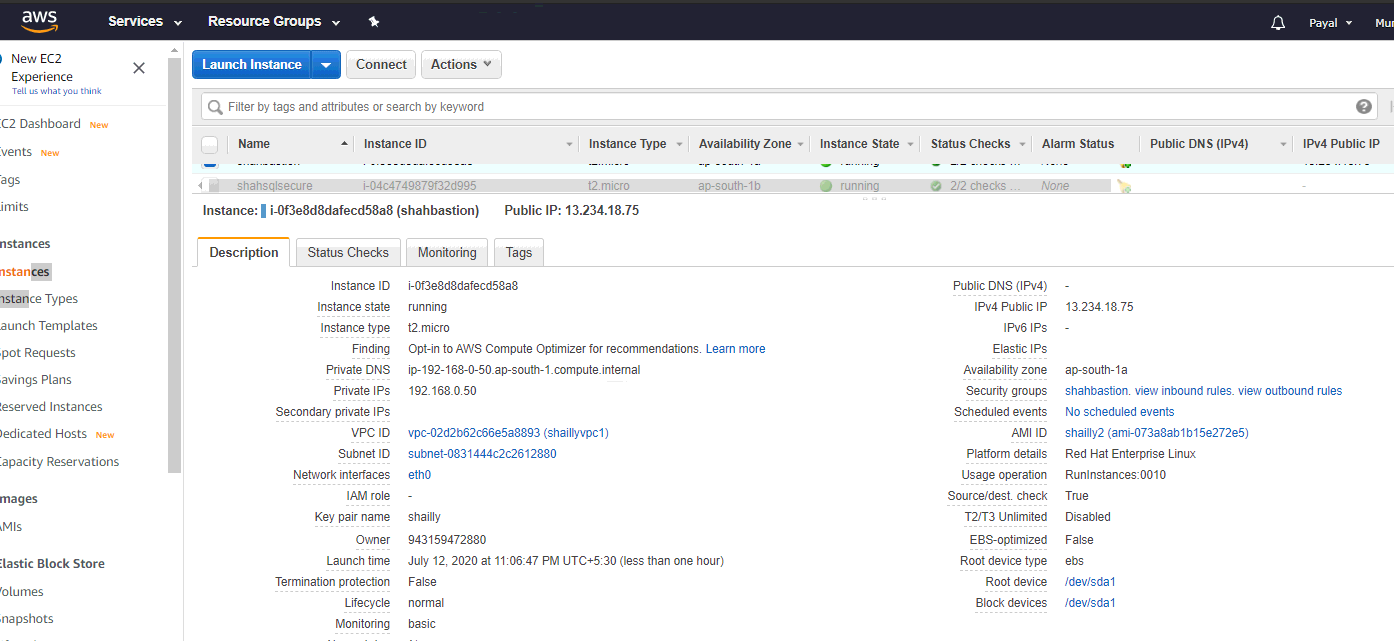
**}**

**}**









Now, login to the bastion host which will be used to the access MysqlOS. First do ssh to login into bastion host, then copy .pem key into bastionhost using winscp software. After this do ssh from bastion host to mysqlos using private key. After all the setup you can do any update in mysqlOS.



Now, you are connected to the MysqlOS(private subnet):-

